2100030336

M. Gayathri Ankitha

.NET PROGRAMMING

LAB-2

IN-LAB:

1. Write a C# code to implement the Tasks on Looping Statements?

TASK1: For a positive integer *n* calculate the *result* value, which is equal to the sum of the odd numbers in *n*

Example

```
n = 1234    result = 4 (1 + 3)
n = 246     result = 0
```

TASK2: For a positive integer *n* calculate the result value, which is equal to the sum of the "1" in the binary representation of *n*.

Example

```
n = 14(decimal) = 1110(binary)        result =  3
n = 128(decimal) = 1000 0000(binary) result  =  1
```

TASK3: For a positive integer *n*, calculate the result value equal to the sum of the first *n* Fibonacci numbers Note: Fibonacci numbers are a series of numbers in which each next number is equal to the sum of the two preceding ones: 0, 1, 1, 2, 3, 5, 8, 13... (F0=0, F1=F2=1, then F(n)=F(n-1)+F(n-2) for n>2)

Example

```
n = 8     result = 33
n = 11    result = 143
```

```
using System;
class Program
{
    static void Main()
    {
        Console.WriteLine("Task 1:");
        Console.Write("Enter an integer (n): ");
        int n = Convert.ToInt32(Console.ReadLine());
        int resultTask1 = CalculateSumOfOddNumbers(n);
        Console.WriteLine($"Result: {resultTask1}");
        Console.WriteLine("\nTask 2:");
        Console.Write("Enter an integer (n): ");
        n = Convert.ToInt32(Console.ReadLine());
        int resultTask2 = CountOnesInBinaryRepresentation(n);
```

```
        Console.WriteLine($"Result: {resultTask2}");
        Console.WriteLine("\nTask 3:");
        Console.Write("Enter an integer (n): ");
        n = Convert.ToInt32(Console.ReadLine());
        int resultTask3 = SumOfFibonacciNumbers(n);
        Console.WriteLine($"Result: {resultTask3}");
    }
    static int CalculateSumOfOddNumbers(int n)
    {
        int result = 0;
        while (n > 0)
        {
            int digit = n % 10;
            if (digit % 2 != 0)
            {
                result += digit;
            }
            n /= 10;
        }
        return result;
    }
    static int CountOnesInBinaryRepresentation(int n)
    {
        int result = 0;
        while (n > 0)
        {
            result += n % 2;
            n /= 2;
        }
        return result;
    }
    static int SumOfFibonacciNumbers(int n)
    {
        int result = 0;
        int a = 0, b = 1;
        for (int i = 0; i < n; i++)
        {
            result += a;
            int temp = a;
            a = b;
            b = temp + b;
        }
        return result;
    }
}
```

2. Write a C# code to implement the Tasks on Arrays?

TASK 1: In a given array of integers ***nums*** swap values of the first and the last array elements, the second and the penultimate etc., if the two exchanged values are even

Example

```
{ 10 , 5, 3, 4}                        =>  {4, 5, 3, 10}
{100, 2, 3, 4, 5}                     => {100, 4, 3, 2, 5}
{100, 2, 3, 45, 33, 8, 4, 54} => {54, 4, 3, 45, 33, 8, 2, 100}
```

TASK 2: In a given array of integers ***nums*** calculate integer ***result*** value, that is equal to the distance between the first and the last entry of the maximum value in the array.

Example

```
{4, 100!, 3, 4}               result = 0
{5, 50!, 50!, 4, 5}          result = 1
{5, 350!, 350, 4, 350!} result = 3
{10!, 10, 10, 10, 10!}   result = 4
```

TASK 3: In a predetermined two-dimensional integer array (square matrix) ***matrix*** insert 0 into elements to the left side of the main diagonal, and 1 into elements to the right side of the diagonal.

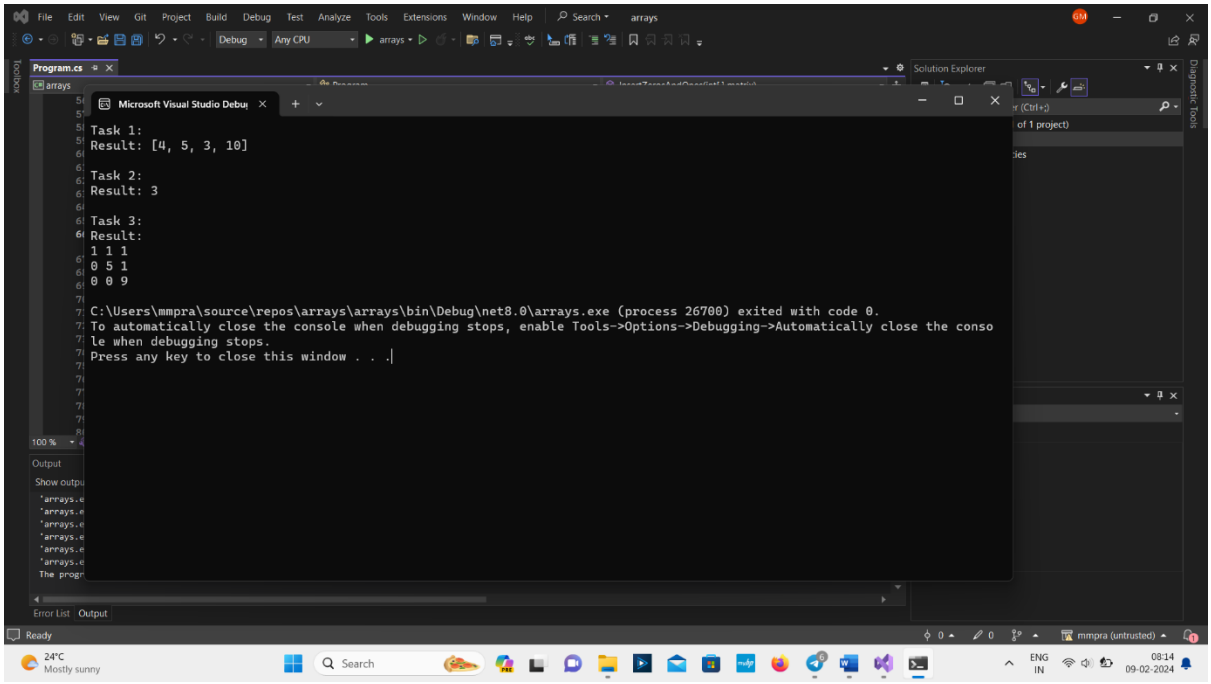
Example

```
{{2, 4, 3, 3},           {{2, 1, 1, 1},
 {5, 7, 8, 5},     =>     {0, 7, 1, 1},
 {2, 4, 3, 3},              {0, 0, 3, 1},
 {5, 7, 8, 5}}              {0, 0, 0, 5}}
```

```
using System;

class Program
{
    static void Main()
    {
        Console.WriteLine("Task 1:");
        int[] numsTask1 = { 10, 5, 3, 4 };
        SwapEvenIndexedValues(numsTask1);
        Console.WriteLine($"Result: [{string.Join(", ", numsTask1)}]");
        Console.WriteLine("\nTask 2:");
        int[] numsTask2 = { 5, 50!, 50!, 4, 5 };
        int resultTask2 = CalculateDistanceBetweenMaxValues(numsTask2);
        Console.WriteLine($"Result: {resultTask2}");
        Console.WriteLine("\nTask 3:");
        int[,] matrixTask3 = {
            { 1, 2, 3 },
            { 4, 5, 6 },
            { 7, 8, 9 }
        };
    InsertZerosAndOnes(matrixTask3);
    Console.WriteLine("Result:");
        PrintMatrix(matrixTask3);
}
static void SwapEvenIndexedValues(int[] nums)
{
    for (int i = 0; i < nums.Length / 2; i += 2)
    {
        if (nums[i] % 2 == 0 && nums[nums.Length - 1 - i] % 2 == 0)
        {
            int temp = nums[i];
```

```
            nums[i] = nums[nums.Length - 1 - i];
            nums[nums.Length - 1 - i] = temp;
        }
    }
}
    static int CalculateDistanceBetweenMaxValues(int[] nums)
    {
        int maxIndex = 0;
        for (int i = 1; i < nums.Length; i++)
        {
            if (nums[i] > nums[maxIndex])
            {
                maxIndex = i;
            }
        }
        return Math.Abs(maxIndex - (nums.Length - 1));
    }
    static void InsertZerosAndOnes(int[,] matrix)
    {
        int n = matrix.GetLength(0);
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
            {
                if (j < i)
                {
                    matrix[i, j] = 0;
                }
                else if (j > i)
                {
                    matrix[i, j] = 1;
                }
            }
        }
    }
    static void PrintMatrix(int[,] matrix)
    {
        int rows = matrix.GetLength(0);
        int cols = matrix.GetLength(1);
        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < cols; j++)
            {
                Console.Write(matrix[i, j] + " ");
            }
            Console.WriteLine();
        }
    }
}
```



3. Write a C# code to implement the Tasks on Functions?

TASK 1: **Create function *IsSorted*, determining whether a given *array* of integer values of arbitrary length is sorted in a given *order* (the order is set up by enum value *SortOrder*). Array and sort order are passed by parameters. Function does not change the array**

TASK 2: Create function ***Transform***, replacing the value of each element of an integer ***array*** with the sum of this element value and its index, only if the given ***array*** is sorted in the given ***order*** (the order is set up by enum value ***SortOrder***). Array and sort order are passed by parameters. To check, if the array is sorted, the function ***IsSorted*** from the Task 1 is called.

Example

```
For {5, 17, 24, 88, 33, 2} and "ascending" sort order values in the array
do not change;
For {15, 10, 3} and "ascending" sort order values in the array do not
change;
For {15, 10, 3} and "descending" sort order the values in the array change
to {15, 11, 5}
```

TASK 3: Create function ***MultArithmeticElements***, which determines the multiplication of a given number of first ***n*** elements of arithmetic progression of real numbers with a given initial element of progression ***a(1)*** and progression step ***t***. a(n) is calculated by the formula a(n+1) = a(n) + t.

Example

```
For a(1) = 5, t = 3, n = 4  multiplication equals to 5*8*11*14 = 6160
```

TASK 4: Create function ***SumGeometricElements***, determining the sum of the first elements of a decreasing geometric progression of real numbers with a given initial element of a progression ***a(1)***) and a given progression step ***t***, while the last element must be greater than a given ***alim***. an is calculated by the formula a(n+1) = a(n) * t, 0<t<1.

Example

```
For a progression, where a(1) = 100, and t = 0.5, the sum of the first
elements, grater than alim = 20, equals to 100+50+25 = 175
```

```
using System;
public enum SortOrder
{
    Ascending,
    Descending
}
class Program
{
    static void Main()
    {
        Console.WriteLine("Task 1:");
        int[] arrTask1 = { 5, 17, 24, 88, 33, 2 };
        Console.WriteLine(IsSorted(arrTask1, SortOrder.Ascending));
        Console.WriteLine(IsSorted(arrTask1, SortOrder.Descending));
        Console.WriteLine("\nTask 2:");
        int[] arrTask2 = { 5, 17, 24, 88, 33, 2 };
        Transform(arrTask2, SortOrder.Ascending);
        Console.WriteLine($"Transformed array: [{string.Join(", ", arrTask2)}]");
        int[] arrTask2Descending = { 15, 10, 3 };
        Transform(arrTask2Descending, SortOrder.Descending);
        Console.WriteLine($"Transformed array: [{string.Join(", ",
arrTask2Descending)}]");
        Console.WriteLine("\nTask 3:");
        double resultTask3 = MultArithmeticElements(5, 3, 4);
        Console.WriteLine($"Result: {resultTask3}");
        Console.WriteLine("\nTask 4:");
        double resultTask4 = SumGeometricElements(100, 0.5, 20);
        Console.WriteLine($"Result: {resultTask4}");
    }
    static bool IsSorted(int[] arr, SortOrder order)
    {
        if (order == SortOrder.Ascending)
        {
            for (int i = 1; i < arr.Length; i++)
            {
                if (arr[i] < arr[i - 1])
                    return false;
            }
        }
        else
        {
            for (int i = 1; i < arr.Length; i++)
            {
                if (arr[i] > arr[i - 1])
                    return false;
            }
        }
        return true;
    }
    static void Transform(int[] arr, SortOrder order)
    {
        if (IsSorted(arr, order))
```

```
        {
            for (int i = 0; i < arr.Length; i++)
            {
                arr[i] += i;
            }
        }
    }
    static double MultArithmeticElements(double a1, double t, int n)
    {
        double result = 1;
        double currentElement = a1;

        for (int i = 0; i < n; i++)
        {
            result *= currentElement;
            currentElement += t;
        }

        return result;
    }
    static double SumGeometricElements(double a1, double t, double alim)
    {
        double result = 0;
        double currentElement = a1;

        while (currentElement > alim)
        {
            result += currentElement;
            currentElement *= t;
        }
        return result;
    }
}
```